ASG
Software Solutions

ASG-DataManager™ IDMS/R Interface

Version 2.5

Publication Number: DMR0200-25-IDMSR
Publication Date: September 1987

ASG Worldwide Headquarters Naples Florida USA | asg.com | info@asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 239.435.2200 Fax: 239.263.3692 Toll Free: 800.932.5536 (USA only)

ASG Documentation/Product Enhancement Fax Form

Please FAX comments regarding ASG products and/or documentation to (239) 263-3692.

Company Name	Telephone Number	Site ID	Contact name

Product Name/Publication	Version #	Publication Date
Product:		
Publication:		
Tape VOLSER:		

Enhancement Request:

ASG Support Numbers

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours. To expedite response time, please follow these procedures.

Please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely or displayed
- A description of the specific steps that immediately preceded the problem
- The severity code (ASG Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)
- Verify whether you received an ASG Service Pack for this product. It may include information to help you resolve questions regarding installation of this ASG product. The Service Pack instructions are in a text file on the distribution media included with the Service Pack.

If You Receive a Voice Mail Message:

1. Follow the instructions to report a production-down or critical problem.
2. Leave a detailed message including your name and phone number. A Support representative will be paged and will return your call as soon as possible.
3. Please have the information described above ready for when you are contacted by the Support representative.

Severity Codes and Expected Support Response Times

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Business Hours Support

Your Location	Phone	Fax	E-mail
United States and Canada	800.354.3578	239.263.2883	support@asg.com
Australia	61.2.9460.0411	61.2.9460.0280	support.au@asg.com
England	44.1727.736305	44.1727.812018	support.uk@asg.com
France	33.141.028590	33.141.028589	support.fr@asg.com
Germany	49.89.45716.222	49.89.45716.400	support.de@asg.com
Singapore	65.6332.2922	65.6337.7228	support.sg@asg.com
All other countries:	1.239.435.2200		support@asg.com

Non-Business Hours - Emergency Support

Your Location	Phone	Your Location	Phone
United States and Canada	800.354.3578		
Asia	65.6332.2922	**Japan/Telecom**	0041.800.9932.5536
Australia	0011.800.9932.5536	**Netherlands**	00.800.3354.3578
Denmark	00.800.9932.5536	**New Zealand**	00.800.9932.5536
France	00.800.3354.3578	**Singapore**	001.800.3354.3578
Germany	00.800.3354.3578	**South Korea**	001.800.9932.5536
Hong Kong	001.800.9932.5536	**Sweden/Telia**	009.800.9932.5536
Ireland	00.800.9932.5536	**Switzerland**	00.800.9932.5536
Israel/Bezeq	014.800.9932.5536	**Thailand**	001.800.9932.5536
Japan/IDC	0061.800.9932.5536	**United Kingdom**	00.800.9932.5536
		All other countries	1.239.435.2200

ASG Web Site

Visit http://www.asg.com, ASG's World Wide Web site.

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/asp/emailproductsuggestions.asp.

If you do not have access to the web, FAX your suggestions to product management at (239) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

Contents

Preface

This *ASG-DataManager IDMS/R Interface* describes the interface facilities between ASG-DataManager (herein called DataManager) and IDMS/R, which enable the user to include IDMS data definitions in the data dictionary, and to produce Schema, Subschema, and Device Media Control Language specifications for direct input to IDMS utilities.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on the DataManager product.

About this Publication

The *ASG-DataManager IDMS/R Interface* consists of these chapters:

- Chapter 1 "Introduction to DataManager IDMS Facilities"
- Chapter 2 "Processing Definitions of an IDMS Database"
- Chapter 3 "IDMS Databases and DataManager"
- Chapter 4 "IDMS Member Types"
- Chapter 5 "DataManager Correspondence Tables"

Publication Conventions

ASG's technical publications use these conventions:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
`lowercase italic monospace`	Information that you provide according to your particular situation. For example, you would replace *`filename`* with the actual name of the file.
`Monospace`	Characters you must type exactly as they are shown. Code, JCL, file listings, or command/statement syntax. Also used for denoting brief examples in a paragraph.

Requesting Publication Changes

Customers and other ASG departments can use a Documentation Correction/Enhancement Request Form to request corrections, updates, and enhancements to publications. The form is included in the front matter of each publication. Forms are also available from the Vice President of Technical Publications.

The Vice President of Technical Publications evaluates requests for documentation changes.

1 Introduction to DataManager IDMS Facilities

ASG-Manager Products and IDMS/R Interface Facilities

For the IDMS user, these interface facilities between DataManager and IDMS/R are provided by ASG-Manager Products (herein called Manager Products):

- The ability to define the IDMS database in a ASG-Manager Products dictionary, to hold the definitions in the dictionary and to process them using ASG-Manager Products commands.
- The ability to define processing views of the IDMS database (that is, the Subschema), to hold the definitions in a ASG-Manager Products dictionary and to process them using ASG-Manager Products commands.

If DictionaryManager's Corporate Dictionary Export for IDD/IDMS is installed, you can also:

- Translate definitions from the ASG-Manager Products dictionary into input statements for the IDD DDDL Compiler.
- Generate input statements for the IDMS Schema, Subschema and DMCL Compilers from members.

And if an ASG-Manager Products Source Language Generation facility is installed, you can PRODUCE record layouts and COBOL, BAL, and PL/I programming source language data descriptions from IDMS-RECORD and IDMS-VIEW member types.

The ability to define an IDMS database demands a further four member types in DataManager. These are IDMS-DATABASE, IDMS-AREA, IDMS-SET, and IDMS-RECORD. IDMS-AREA and IDMS-RECORD are used to define the physical contents of the database; IDMS-SET is used to define how records are linked in the database.

Additionally, two further member types, IDMS-VIEW and IDMS-SUBSCHEMA, together with facilities at the MODULE, PROGRAM and SYSTEM data definition levels, are required to allow the processing view of the database to be defined.

IDMS-LOGICAL-RECORD provides the ability to define logical records which may be accessed by programs when using processing views of the IDMSDATABASE (e.g., Subschema). IDMS-PATH-GROUP provides the ability to define the paths used by programs via the IDMS Logical Record facility (LRF) when accessing these logical records.

DataManager IDMS member types are documented in Chapter 4, "IDMS Member Types," on page 17.

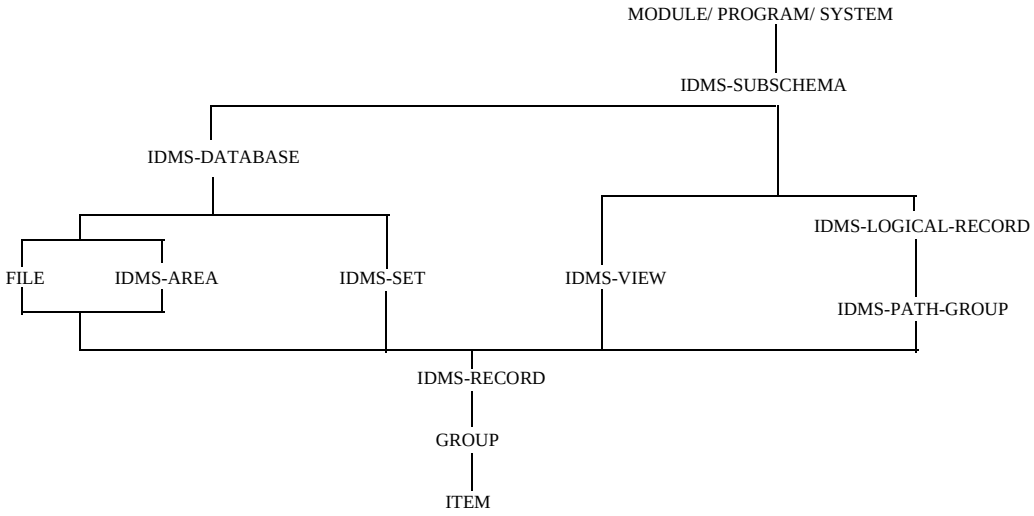


Figure 1. IDMS/R Interface Member Types in the DataManager Hierarchy

Source Language Generation

The DataManager Source Language Generation facilities SLl, SL2, and SL3 can be used to produce record layouts and/or COBOL, PL/I, or BAL (respectively) programming source language data descriptions from IDMS-RECORD and IDMS-VIEW member types.

The basic form of the PRODUCE command is described in the publication *ASG-Manager Products Source Language Generation.*

The IDMS-RECORD and IDMS-VIEW member appears in the output as a top level GROUP, containing any members listed in the IDMS-RECORD CONTAINS clause or the IDMS-VIEW SELECTING clause.

The tailoring macros that apply to the generation of record layouts and COBOL, PL/I, or BAL data descriptions are defined in the publication *ASG-Manager Products Source Language Generation*.

Note that these generation control options:

GIVING { FD-ONLY | RECORDS-ONLY | ALL-FILE }

are not relevant when member-name is an IDMS-RECORD or IDMS-VIEW.

This table tabulates ASG-Manager Products member types against the IDD/IDMS Compilers to which they can generate input.

	IDD/IDMS Compiler				
Manager Member Type	**DD DL**	**Schema DDL**	**Subschema DDL**	**DMCL**	**COBOL PL1 BAL**
ITEM	X	X			X
GROUP	X	X			X
USERVIEW	X				
ENTITY	X				
VIEWSET	X				
IDMS-RECORD	X	X			X
IDMS-AREA		X			X
IDMS-SET		X			
IDMS-DATABASE		X			
IDMS-SUBSCHEMA			X	X	
IDMS-VIEW			X		X
IDMS-LOGICAL-RECORD			X		
IDMS-PATH-GROUP			X		

2 Processing Definitions of an IDMS Database

Processing IDMS Members

In order that the definitions of IDMS databases may be processed by ASG-Manager Products in the same way as other members of the data dictionary, the interrogate keywords IDMS-DATABASE, IDMS-AREA, IDMS-SET, IDMS-RECORD, IDMSVIEW, IDMS-SUBSCHEMA, IDMS-LOGICAL-RECORD, and IDMS-PATHGROUP are added to the member-type keywords available for use in these commands:

- BULK
- GLOSSARY
- LIST
- PERFORM
- REPORT
- WHICH

In addition some keywords are available with these command variants:

- VIA clause of the WHICH and WHAT commands
- FOR clause of the GLOSSARY command
- HAS/HAVE clause of the WHICH command.

Interrogate Keywords Available with the VIA Clause of the WHICH and WHAT Commands

The following table shows the interrogation keywords available with VIA and the clauses to which each of these keywords refer. The clauses and keywords are grouped according to member type.

The CALLS, SORT-KEYS, and KEYS keywords, when used with the IDMS Interface, refer to clauses that are specific to IDMS, as well as those available with the DataManager Nucleus. The other keywords are additional to those available with the DataManager Nucleus. The ACCESSES keyword can also be used with the ADABAS Interface and the System 2000/80 Interface, and the KEYS keyword also refers to clauses that are specific to System 2000/80.

Interrogate Keywords available with the VIA clause of the WHICH and WHAT Commands

Member Type	Clause	Interrogation
FILE	SORT-KEY	VIA SORT-KEYS
	SORT-KEY	VIA SORTED
GROUP	KEYS	VIA KEYS
IDMS-AREA	CALL	VIA CALLS
	IN	VIA IN
IDMS-DATABASE	AREA	VIA AREAS
	DATASET	VIA DATASETS
	DATASET	VIA JOURNAL
	SETS	VIA SETS
	STAND-ALONE	VIA STAND-ALONE
IDMS-LOGICAL-RECORD	CONTAINS	VIA CONTAINS
	ERASE	VIA ERASE
	MODIFY	VIA MODIFY
	OBTAINS	VIA OBTAINS
	STORE	VIA STORE
IDMS-PATH-GROUP	COMPUTE	VIA COMPUTE
	CONNECT	VIA CONNECT
	CONNECT-TO	VIA CONNECT-TO
	DISCONNECT	VIA DISCONNECT
	DISCONNECT-FROM	VIA DISCONNECT-FROM
	ERASE	VIA ERASE
	EVALUATE	VIA EVALUATE
	FIND	VIA FIND
	FIND-USING	VIA FIND-USING
	GET	VIA GET
	FOR	VIA FOR

Interrogate Keywords available with the VIA clause of the WHICH and WHAT Commands

Member Type	Clause	Interrogation
	KEEP	VIA KEEP
	MODIFY	VIA MODIFY
	PATHS	VIA PATHS
	STORE	VIA STORE
	WITHIN	VIA WITHIN
IDMS-RECORD	AREA	VIA AREAS
	CALL	VIA CALLS
	CONTAINS	VIA CONTAINS
	IF	VIA IF
	STORED USING KEY	VIA KEYS
IDMS-SET	MEMBER	VIA MEMBER
	OWNER	VIA OWNER
	OWNED-BY	VIA OWNER-AREA
	SORTED	VIA SORT-KEYS
	SORTED	VIA SORTED
	STORED-VIA	VIA STORED-VIA
	VSAM-FILE	VIA VSAM-FILE
IDMS-SUBSCHEMA	ACCESSES	VIA ACCESSES
	AREA †	VIA AREAS
	AS ‡	VIA AS
	TAPE-FILE / FILE	VIA DATASETS
	TAPE-FILE / FILE	VIA JOURNAL
	LOGICAL-RECORD	VIA LOGICAL-RECORD
	RECORD	VIA RECORD
	SELECTING	VIA SELECTING
	SET	VIA SETS
	STATISTICS-FOR	VIA STATISTICS-FOR
	STATISTICS-OF	VIA STATISTICS-OF
	STATISTICS-TO	VIA STATISTICS-TO

Interrogate Keywords available with the VIA clause of the WHICH and WHAT Commands

Member Type	Clause	Interrogation
IDMS-AREA	RECORD	VIA RECORD
	SELECTING	VIA SELECTING
MODULE	CALLS	VIA CALLS
PROGRAM	CALLS	VIA CALLS
SYSTEM	CALLS	VIA CALLS

† VIA AREAS also refers to the AREA subordinate clause of the DMCL clause in IDMS-SUBSCHEMA.

‡ VIA AS refers to the AS subordinate clause of the DMCL clause in IDMS-SUBSCHEMA.

Interrogate Keywords Available with the FOR Clause of the GLOSSARY Command and the HAS/HAVE Clause of the WHICH Command

This table shows the keywords available for each IDMS Member Type in the command:

`GLOSSARY FOR` *`member-type`* `GIVING` *`clauses`*

and in the attribute-specification on the WHICH HAS/HAVE command.

IDMS-DATABASE	IDMS-SET	IDMS-SUBSCHEMA	IDMS-RECORD
*DATASET	ORDER	ACCESSES	IDENTITY
*JOURNAL	*MODE	*ALL-AREAS	*STORED
*AREAS	*OWNER	*AREA	*USING
*SETS	*OWNED-BY	*RECORD	DUPLICATES
*STAND-ALONE	*MEMBER	*ALL-SETS	*VSAM
RECORD-ID-START		*SET	*KEY
MAXIMUM-RECORDS		*DMCL	*VIA
-PER-PAGE		ASSIGN	*AREA
PAGE-GROUP		DEVICE	*MINIMUM-ROOT
		*DISKS	*MINUMUM-
		*VIEW	FRAGMENT
		AUTHORIZATION	*CALL
		USAGE	STORAGE

IDMS-DATABASE	IDMS-SET	IDMS-SUBSCHEMA	IDMS-RECORD
		*STATISTICS	OCCURENCES
		LR-CURRENCY	*SELECTING
		*LOGICAL-RECORD	

This table shows the interrogate keywords available with GLOSSARY (FOR Clause) and WHICH (HAS/HAVE Clause):

IDMS-AREA	IDMS-LOGICAL-RECORD	IDMS-PATH-GROUP	IDMS-VIEW
*PAGES	*CONTAINS	*FOR-SELECTION	RECORD
*CALL	ERROR	*DO	*SELECTING
	NOT-FOUND		
	*OBTAIN		
	*MODIFY		
	*STORE		
	*ERASE		

Notes:

1. All keywords can be used with the GLOSSARY command.

2. With the WHICH command:

   - All keywords can be used with the SPECIFIED keyword
   - The keywords marked with an asterisk (*) cannot be used with an operator, that is, EQ, NE, etc.

3 IDMS Databases and DataManager

Defining the IDMS Database

The IDMS database can be viewed as a collection of record occurrences grouped into processing areas within files, or into files within processing areas. Records exist as physical entities accessible in their own right, but also in logical sets whereby access to a specific record causes the accessing of a set of logically related records. In IDMS terminology, the rules that govern both the physical and logical aspects of a database are called the *Schema*.

Consider, for example, the database system in Figure 2 on page 11: an unbroken line represents a RECORD to AREA relationship; a broken line represents a RECORD to SET relationship.

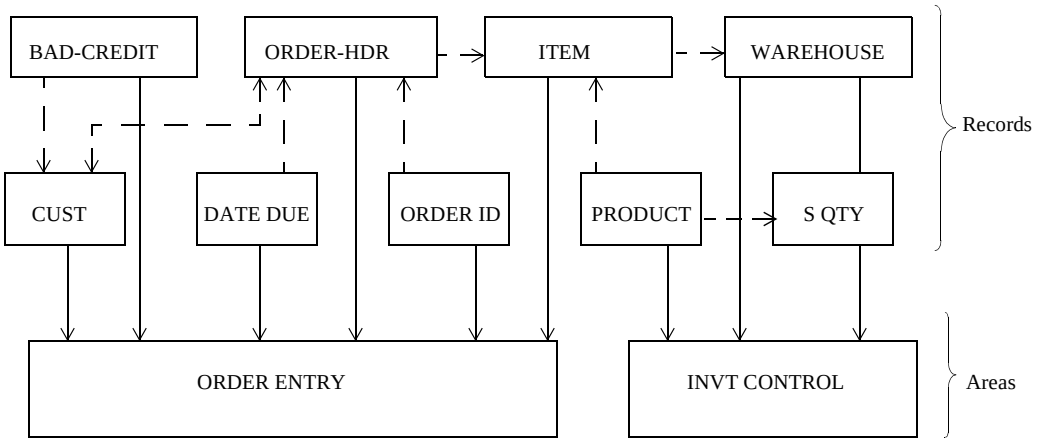


Figure 2. Defining an IDMS Database

A more conventional representation of the record sets could be as shown in Figure 3 on page 12.

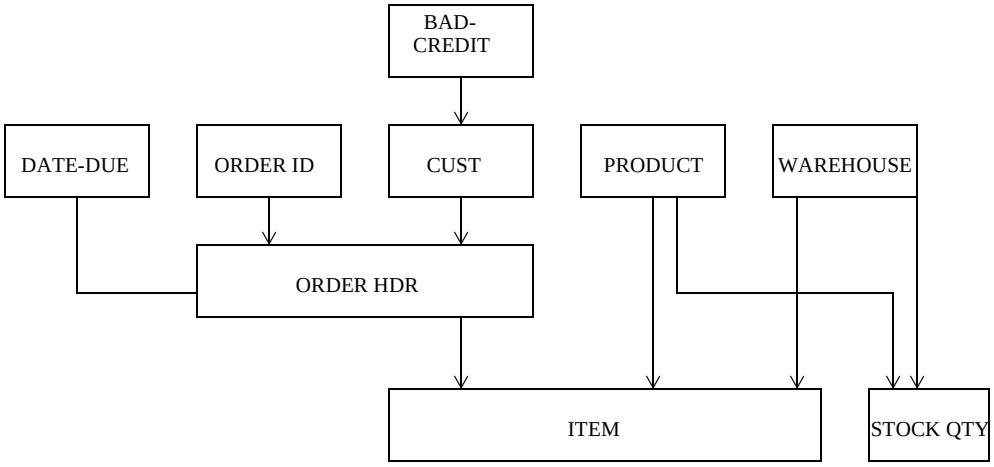


Figure 3. Defining an IDMS Database

For DataManager the aforementioned database would be represented as an IDMS-DATABASE member containing seven IDMS-SET members, and nine IDMS-RECORD members. The IDMS-DATABASE member would specify the names of all appropriate area subdivisions and sets, together with physical file details. The IDMS-AREA members would define the physical relationships of files to areas. The IDMS-SET members would define the set relationships determined for the database. The IDMS-RECORD members would hold details for each record type. The data items which constitute the records would be defined via the normal DataManager GROUP and ITEM data definition statements.

Access and Manipulation of Data

Any given program (run unit) that accesses the database is unlikely to access all record types and perhaps only requires certain data items from certain records. These requirements can be defined in an IDMS-VIEW member. Furthermore, a given program may have restrictions placed on it, in that it may read only certain data items, update other data items, etc. The DataManager IDMS-SUBSCHEMA data definition statement is used to define which database, areas, sets, views, records, and data items are processed, together with any sensitivity details. The data definitions of MODULE, PROGRAM, and/or SYSTEM specify which subschemas are processed. If the IDMS Logical Record facility is to be used, the IDMS LOGICAL-RECORD and IDMS-PATH-GROUP data definition statements are used to define the logical records and the paths required to obtain the data.

The data definitions can be inserted into the data dictionary's source dataset by INSERT or ADD commands, in the same way as any other DataManager data definitions.

Naming Conventions

DataManager member names have a maximum permitted length of 32 characters. However, in IDMS, the names of some entities (database, subschema, and DMCL) are restricted to a maximum of 8 characters, while others (files, areas, records, sets, and buffers) are restricted to 16 characters.

Thus, DataManager member names can be used as IDMS names if they contain no more than the maximum number of characters allowed; but if advantage is taken of the greater length permitted for DataManager member names, other means must be used for defining the names to be used in IDMS contexts. DataManager provides two ways in which this can be achieved: the ALIAS clause and the KNOWN-AS clause.

The ALIAS clause is available in the data definition syntax of any member type. It can be used to state an IDMS specific alias, which can be used instead of the member name when Source Language Generation is performed from the member, or when the DictionaryManager Corporate Dictionary Export for IDD/IDMS is used.

The KNOWN-AS clause is available in the data definition syntax of IDMS-AREA, IDMS-RECORD, IDMS-LOGICAL-RECORD, IDMS-VIEW, and IDMS-SUBSCHEMA members, in GROUP members, and in the PROCESSES IDMS clause of MODULE, PROGRAM, and SYSTEM members. The KNOWN-AS clause declares a local name for a contained or otherwise referenced member; this local name can be used in place of the contained or referenced member's name when Source Language Generation is performed from the containing or referring member, or when the DictionaryManager Corporate Dictionary Export for IDD/IDMS is used.

Examples of IDMS Member Types

Example of an IDMS-DATABASE Data Definition

```
IDMS-DATABASE
DATASET ORDER-FILE ASSIGN ORDENT
DATASET INVENTORY-FILE ASSIGN INVTCON
AREAS ORDER-ENTRY,INVT-CONTROL
SETS     BAD-CREDIT-SET, CUST-ORDER-SET, DATE-ORDER-SET,
         ORDER-SET ,ORDER-ITEM-SET ,WAREHOUSE-SET ,PRODUCT-SET
RECORD-ID-START 50
MAXIMUM-RECORDS-PER-PAGE 100
PAGE-GROUP 10
DESCRIPTION  'A SAMPLE IDMS SCHEMA'
             'SPECIFICATION'
NOTE     'SCHEMA EXAMPLE IS IN CULLIMANE'
         'CORP IDMS DATABASE DESIGN'
         'AND DEFINITION GUIDE'
;
```

Example of an IDMS-AREA Data Definition

```
IDMS-AREA
PAGES 20001  TO 20100
IN ORDER-FILE BLOCK 1 TO 100
;
```

Example of an IDMS-SET Data Definition

```
IDMS-SET
ORDER SORTED
MODE CHAIN  LINKED-PRIOR
OWNER BAD-CREDIT-RCD  NEXT-POSITION 1
                      PRIOR-POSITION 2
MEMBER CUST-RCD NEXT-POSITION 1
               PRIOR-POSITION 2
               OWNER-POSITION 5
               INDEX-POSITION AUTO
MANDATORY MANUAL
SORT-KEY DBKEY DUPLICATES FIRST
;
```

Example of an IDMS-RECORD Data Definition

```
IDMS-RECORD
IDENTITY 202
STORED BY KEY CUST-ID
        DUPLICATES DISALLOWED
VSAM-TYPE FIXED LENGTH SPANNED
AREA ORDER-ENTRY
STORAGE STORE-STR-1
OCCURRENCES 100
CONTAINS CUST-ID KNOWN-AS CI,
        CUST-NAME KNOWN-AS CN,
        CUST-ADDRESS KNOWN-AS CA,
        CUST-CREDIT KNOWN-AS CC,
        CUST-SALES-INFO KNOWN-AS CS
;
```

Example of an IDMS-LOGICAL-RECORD Data Definition

```
IDMS-LOGICAL-RECORD
CONTAINS REC3 KNOWN AS REC3ROLE
        , REC80
        , REC84
ERROR NOCLEAR
NOT-FOUND CLEAR
OBTAIN PATHS OPG1, OPG2
MODIFY PATHS MPG1
ERASE PATHS EPG1 ,  EPG2
STORE PATHS SPG1 ,  SPG2
;
```

Example of an IDMS-PATH-GROUP Data Definition

```
IDMS-PATH
FOR KEYWORD KEYNAME2
DO COMPUTE 187 EQ  'STRINGX'
DO CONNECT REC83 TO SET53
DO IF NOT SET SET55 EMPTY ON 4 RETURN 5
DO FIND KEEP EXCLUSIVE
        CALC FIRST RECORD REC1 KEY
        FIELD ITEM1 IN GROUP1 OF REC1LR
AND
        FIELD ITEM2 EQ ARITHMETIC 'ASTRING1' CON ITEM3
AND NOT
        'ASTRING2' LT FIELD ITEM4 IN GROUP2 OF REC1 LR
;
```

Example of an IDMS-VIEW Data Definition

```
IDMS-VIEW
RECORD CUST-RCD
        SELECTING CUST-NAME KNOWN-AS CN,
        CUST-CREDIT KNOWN-AS CC
;
```

Example of an IDMS-SUBSCHEMA Data Definition

```
IDMS-SUBSCHEMA
ACCESSES SAMSTMC
AUTHORIZATION ON
USAGE MIXED
LR-CURRENCY RESET
AREA INVT-CONTROL OPTIONS ALLOW RETRIEVAL
SET WHSE-QTY-SET
SET PROD-QTY-SET
RECORD WAREHOUSE-RCD
RECORD PRODUCT-RCT OPTIONS DISALLOW ERASE
VIEW CUSTVIEW OPTIONS DISALLOW MODIFY
DMCL SAMDINVT 1-0 BUFFER INVTBUFF PAGES 8 SIZE 3156
        AREA INVT-CONTROL
NOTE  'DMCL AND SUBSCHEMA DETAILS'
;
```

Example of an IDMS PROCESSES Definition

```
MODULE
PROCESSES IDMS SUBSCHEMA SAMBVINT
;
```

4 IDMS Member Types

Introduction

To enable an IDMS environment to be fully reflected in the data dictionary maintained by DataManager, the DataManager IDMS Interface provides eight additional member types:

- IDMS-DATABASE
- IDMS-AREA
- IDMS-SET
- IDMS-RECORD
- IDMS-LOGICAL-RECORD
- IDMS-PATH-GROUP
- IDMS-VIEW
- IDMS-SUBSCHEMA

DataManager also provides an extension to the MODULE, PROGRAM, and SYSTEM data definition statements, to reflect the processing view of the database.

The IDMS-DATABASE Member Type

Syntax of the IDMS-DATABASE Member Type

```
IDMS-DATABASE

DATASET file-name  [ASSIGN external-name]  [DEVICE device]
[DATASET file-name  [ASSIGN external-name]  [DEVICE device]]...

[JOURNAL ASSIGN external-name [DEVICE device]]

AREAS area-name [,area-name]. . .

SETS sets-name [,sets-name]. . .

[STAND-ALONE record-name [,record-name]...]

[RECORD-ID-START i]

[MAXIMUM-RECORDS-PER-PAGE j]

[PAGE-GROUP k]

[common-clauses]

{ ; }
{ . }
```

where:

file-name is the name of a FILE member that defines a physical IDMS file that holds some or all of the database being defined

external-name is the logical file name used in job control statements to indicate the external dataset name (physical file name) of the file identified in the DATASET clause

device is:

```
{ VSAM    }
{ KSDS    }
{ ESDS    }
{ RRDS    }
{ PATH    }
{ device-a }
```

device-a is any valid device-type (undelimited character string)

area-name is the name of a member that is an IDMS-AREA

set-name is the name of a member that is an IDMS-SET

record-name is the name of a member that is an IDMS-RECORD

i is an integer in the range 10 to 9999

j is an integer in the range 3 to 4095

k is an integer in the range 1 to 32767

common-clauses are described in the *ASG-Manager Products Dictionary/Repository User's Guide*.

The IDMS-DATABASE Data Definition

The IDMS-DATABASE data definition defines an IDMS Schema. All IDMS rules for Schema definition must be complied with.

Common clauses can be present in any type of data definition statement. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

Clauses can be declared in any order, provided that subordinate clauses are not separated from the other elements of the clause of which they form a part. (The DATASET clause includes its subordinate ASSIGN and DEVICE clauses, and no other clause must be interposed between its subordinate clauses.)

IDMS-DATABASE Dummy Records

A record containing the database data definition statement can be inserted into the data dictionary source dataset by a suitable command, and an encoded record can subsequently be generated and inserted into the data entries dataset. However, if when the encoded record is generated, any member whose name appears in the database data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as:

- A dummy FILE if the member name appears in a DATASET clause
- A dummy IDMS-AREA if the member name appears in the AREAS clause
- A dummy IDMS-SET if the member name appears in the SETS clause
- A dummy IDMS-RECORD if the member name appears in the STAND- ALONE clause
- A dummy ITEM if the member name appears in a SEE clause

The DEVICE and AREAS Clauses

The DEVICE clause is required for DOS disk files, but is optional for OS files. For DOS disk files, the device type must be one valid for IDMS.

The AREAS clause specifies all areas within the files of this database.

The SETS and STAND-ALONE Clauses

The SETS clause specifies all sets within the IDMS database, and thus indirectly (via the IDMS-SET members referenced) specifies all records that are owners and members of these sets. The only other records in the database are those specified in the STAND-ALONE clause.

The STAND-ALONE clause is used to specify all those records that are not included in any set, that is, records that are stand-alone records in the database.

The RECORD-ID-START, MAXIMUM-RECORDS-PER-PAGE and PAGE-GROUP Clauses

The RECORD-ID-START clause specifies the start number that the schema compiler uses when numbering schema records.

The MAXIMUM-RECORDS-PER-PAGE clause specifies the maximum numbers of records that can be stored on one page in the database.

The PAGE-GROUP clause specifies the page group that contains the schema areas.

The IDMS-AREA Member Type

Syntax of the IDMS-AREA Member Type

```
IDMS-AREA

PAGES page-1 { TO  } page-2
             { FOR }
   { OF } file-name BLOCK block [  TO block  ]
   { IN }                       FOR {ALL}
                                    { k }
  [{ OF } file-name BLOCK block [  TO block  ]]...
   { IN }                       FOR {ALL}
                                    { k }

[CALL process-name [KNOWN AS procedure-name]{  BEFORE}
                                            { AFTER }
                                            { ERROR }

[ { READY [FOR{ EXCLUSIVE } [{ UPDATE    } ] } ] ]]...
  {           { PROTECTED }  { RETRIEVAL } }   }
  {           { SHARED    }                }   }
  {           { UPDATE    }                }   }
  {           { RETRIEVAL }                }   }
  { FINISH                                 }   }
  { COMMIT                                 }   }
  { ROLLBACK                               }   }

[common-clauses]

{ ; }
{ . }
```

where:

page-1 is an integer in the range 1 to 1073741821

page-2 is an integer in the range 2 to 1073741822

file-name is the name of a FILE member identifying a physical IDMS file of which the area being defined is a part

block is an integer in the range 1 to 2147483647K is an integer in the range 1 to 1073741822

process-name is the name of a member that is a SYSTEM, PROGRAM, or MODULE

`procedure-name` is a name not more than eight characters in length, that both conforms to the rules for member names and is a legal CSECT name or entry point for input into IBM OS or DOS assemblers. The procedure-name can be used instead of the process-name, member name, or alias when IDMS source language is generated from this data definition by the DataManager Source Language Generation facility. The procedure-name is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for procedure-name when the data definition in which it appears is encoded) so procedure-name cannot be interrogated and can be the same as another name, an alias or a catalog classification in the data dictionary. The procedure-name is the name by which process-name is known only in the context of this data definition

`common-clauses` are described in the *ASG-Manager Products Dictionary/Repository User's Guide*.

The IDMS-AREA Data Definition

This data definition statement allows subdivisions of database files to be defined.

All IDMS rules for area definition must be complied with in the IDMS-AREA data definition.

At least one OF or IN file-name clause must be present, specifying the dataset of which the area is a part. An area may spread over several datasets, each of which is specified in an OF or IN file-name clause.

As many CALL clauses as desired may be specified.

Other keywords have the same meaning as in IDMS.

Common clauses can be present in any type of data definition statement. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

Clauses can be declared in any order, provided that subordinate clauses are not separated from the other elements of the clause of which they form a part. (For example, the OF or IN clause includes its subordinate BLOCK and TO clauses, and no other clause must be interposed between its subordinate clauses, and the OF or IN clause is itself subordinate to the PAGES clause, from which it must not be separated.)

IDMS-AREA Dummy Records

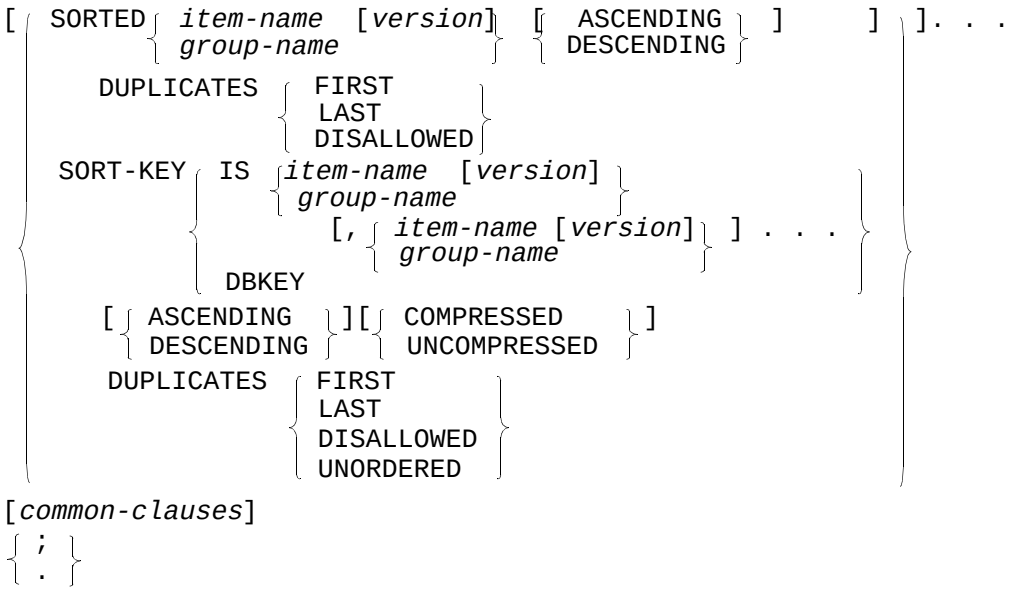
A record containing the area data definition statement can be inserted into the data dictionary source dataset by a suitable command. An encoded record can subsequently be generated and inserted into the data entries dataset. However if when the encoded record is generated, any member whose name appears in the area data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as:

- A dummy FILE if the member name appears on the OF or IN clause
- A dummy MODULE if the member name appears in a CALL clause
- A dummy ITEM if the member name appears in a SEE clause

The IDMS-SET Member Type

Syntax of the IDMS-SET Member Type

```
IDMS-SET
[ORDER { FIRST  }]
       { LAST   }
       { NEXT   }
       { PRIOR  }
       { SORTED }

[{ LINKED-PRIOR                                          }]
{ MODE { CHAIN  [ LINKED-PRIOR ]                         }}
{      { VSAM INDEX file-name                            }}
{      { INDEXED KEYS p [DISPLACED page-count]           }}

[{ OWNER record-name [NEXT-POSITION{ position }  ]       }]
{                                  { AUTO     }          }
{                    [PRIOR-POSITION{ position }  ]      }
{                                  { AUTO     }          }
{ OWNED-BY SYSTEM AREA area-name                         }
{    {FROM page-1 TO page-2                              }
{    {OFFSET { PERCENT  i  }    FOR   {PERCENT  s  }     }
{       {PAGES page-3 }          { PAGES page-4 }       }}

[MEMBER record-name [NEXT-POSITION{ position }]
                                  { AUTO     }

   [PRIOR-POSITION{ position }][OWNER-POSITION{ position }]
                  { AUTO     }                { AUTO     }
   [INDEX-POSITION{ position }]
                  { AUTO     }
   [ { MANDATORY }][ { AUTOMATIC }]
     { OPTIONAL  }   { MANUAL    }
```



where:

file-name is the name of a FILE member

p is an integer in the range 3 to 8180

page-count is an integer in the range 0 to 32767

record-name is the name of a member that is an IDMS-RECORD

position is an integer in the range 1 to 8180

area-name is the name of a member that is an IDMS-AREA

page-1 and *page-2* are integers in the range 1 to 1073741822

i is an integer in the range 0 to 99

s is an integer in the range 1 to 100

page-3 is an integer in the range 1 to 1073741821

page-4 is an integer in the range 2 to 1073741822

item-name is the name of a member that is an ITEM

group-name is the name of a member that is a GROUP

version is an unsigned integer in the range 1 to 15, being a number specifying which version of the item is relevant to this definition. The version is within the HELD-AS form, or within a defaulted form. If version is omitted, a default value of 1 is assumed.

`common-clauses` are as described in the *ASG-Manager Products Dictionary/Repository User's Guide*.

The IDMS-SET Data Definition

This data definition statement allows the logical connection between record types to be defined.

Each IDMS-SET member defines a logical relationship between two or more record types. All IDMS rules for Set definition must be obeyed.

The ORDER clause defines the point of insertion or retrieval of records within the defined set.

LINKED-PRIOR indicates that prior pointer positions are to be assigned within this set.

The MODE clause defines how the pointers are to be maintained.

The CHAIN clause links each record in the set to the next; if present, the LINKED-PRIOR clause specifies that each record in the set is also linked to the prior one.

The VSAM INDEX clause identifies this set as a native VSAM one.

The INDEXED KEYS clause identifies this set as an indexed one; if present, the DISPLACED clause specifies how far from their owners the internal index records are to be stored.

The OWNER clause names the IDMS-RECORD member that defines the record type that is the owner of this set.

The NEXT-POSITION clause and, if present, the PRIOR-POSITION clause define key positions determined according to IDMS convention.

The MEMBER clauses name the IDMS-RECORD members that define the record types that are members of this set. The keywords in the MEMBER clause have the same meaning as in IDMS.

The OWNER-POSITION subordinate clause must be included if occurrences of this record type will carry an OWNER pointer overhead within this set.

If the SORTED subordinate clause is present, its subordinate DUPLICATES clause must be present.

Common clauses can be present in any type of data definition statement. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

Clauses can be declared in any order, provided that subordinate clauses are not separated from the other elements of the clause of which they form a part. (For example, the MEMBER clause includes all its subordinate keywords and clauses, and no other clause must be interposed between these.)

IDMS-SET Dummy Records

A record containing the IDMS set data definition statement can be inserted into the data dictionary source dataset by a suitable command and an encoded record can subsequently be generated and inserted into the data entries dataset. If, when the encoded record is generated, any member whose name appears in the IDMS set data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The type of dummy created depends upon the clause in which the member name appears:

- A dummy IDMS-RECORD if its name appears in an OWNER or MEMBER clause
- A dummy FILE if its name appears in a MODE VSAM INDEX clause
- A dummy IDMS-AREA if its name appears in an OWNED-BY SYSTEM AREA clause
- A dummy ITEM if its name appears in a SORTED, SORT-KEY, or SEE clause

The IDMS-RECORD Member Type

Syntax of the IDMS-RECORD Member Type

```
IDMS-RECORD

[IDENTITY record-id]

[STORED  ( DIRECT
           VIA set-name [DISPLACED page-count]
           { USING } KEY { item-name [version] }
           { BY    }     { group-name          }
                       [,{ item-name [version] } ]...
                         { group-name          }
                       DUPLICATES { FIRST      }
                                  { LAST       }
                                  { DISALLOWED }
           VSAM [FILE file-name
             { USING } KEY { item-name [version] }
             { BY    }     { group-name          }
             DUPLICATES { DISALLOWED }
                        { UNORDERED  }
         )

[VSAM-TYPE { FIXED    } LENGTH { SPANNED   }
           { VARIABLE }        { UNSPANNED }
```

```
[AREA area-name
  [ { FROM page-1 TO page-2                                  } ]]
    { OFFSET { PERCENT i    } FOR { PERCENT k    }           }
             { PAGES page-3 }     { PAGES page-4 }

[MINIMUM-ROOT { CONTROL     } ]
              { root-length }
              { RECORD      }

[MINIMUM-FRAGMENT { fragment-length } ]
                  { RECORD          }

[CALL process-name [KNOWN-AS procedure-name]
  { BEFORE } [ { STORE      }   ]]...
  { AFTER  }   { CONNECT    }
  { ERROR  }   { MODIFY     }
               { DISCONNECT }
               { ERASE      }
               { FIND       }
               { GET        }
               { RETURN     }

[STORAGE string]

[OCCURRENCES p]

[[form] CONTAINS
     content [IF clause][ELSE content][IF clause]]...
   [,content [IF clause][ELSE content][IF clause]...]...]

[common-clauses]
{ ; }
{ . }
```

where:

record-id is an integer in the range 10 to 9999

set-name is the name of a member that is an IDMS-SET

page-count is an integer in the range 0 to 32767

item-name is the name of a member that is an ITEM

group-name is the name of a member that is a GROUP

version is an unsigned integer in the range 1 to 15, being a number specifying which version of the item is relevant to this definition. The version is within the HELD-AS form, or within a defaulted form. If version is omitted, a default value of 1 is assumed.

file-name is the name of a FILE member

area-name is the name of a member that is an IDMS-AREA, that defines the area (of the IDMS database) within which this record resides

page-1 and *page-2* are integers in the range 1 to 1073741822

i is an integer in the range 0 to 99

k is an integer in the range 1 to 100

page-3 is an integer in the range 0 to 1073741821

page-4 is an integer in the range 1 to 1073741822

root-length is an unsigned integer, a multiple of 4, being the number of characters to be included in the initial portion of the record, if the record is variable length

fragment-length is an unsigned integer, a multiple of 4, being the minimum number of characters to be included in subsequent portions of the record, if the record is variable length

process-name is the name of a member that is a SYSTEM, PROGRAM or MODULE

procedure-name is a name not more than eight characters in length, that both conforms to the rules for member names and is a legal CSECT name or entry point for input into IBM OS or DOS assemblers. The procedure-name can be used instead of the *process-name* member name or alias when IDMS source language is generated from this data definition by the DataManager Source Language Generation facility. The *procedure-name* is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for *procedure-name* when the data definition in which it appears is encoded) so *procedure-name* cannot be interrogated and can be the same as another name, an alias or a catalog classification in the data dictionary. The *procedure-name* is the name by which *process-name* is known only in the context of this data definition.

string is up to 16 alphanumeric characters

p is an integer in the range 0 to 2147483647

form is one of these options:

```
{ ALIGNED     }   { ENTERED-AS    }
{ UNALIGNED   }   { HELD-AS       }
{ NOT-ALIGNED }   { REPORTED-AS   }
                  { DEFAULTED-AS  }
```

content declares an item, a subordinate group, or an array, in the format:

```
{ { item-name [version] } [ { ALIGNED     } ][KNOWN-AS local-name]
  { group-name          }   { UNALIGNED   }
                            { NOT-ALIGNED }

{ { integer              } { { item-name[version] } }
  { item-name-a[version] } { { group-name         } }
    [ { ALIGNED     } [KNOWN-AS local-name][INDEXED-BY index-name]
      { UNALIGNED   }
      { NOT-ALIGNED }
```

local-name is a name, conforming to the rules for member names, that can be used instead of the name or alias of the contained member, when record layouts or source language data descriptions are generated from this data definition by the DataManager Source Language Generation facility. The *local-name* is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for *local-name* when the data definition in which it appears is encoded) so *local-name* cannot be interrogated and can be the same as another name, an alias, or a catalog classification in the data dictionary. The *local-name* is the name by which the contained member is known only within the record defined by this data definition.

integer is an unsigned integer of from 1 to 18 digits, being the number of times *item-name* or *group-name* occurs in the array

item-name-a is the name of a member that is an ITEM. This form of array declaration signifies that when the record here defined is processed by an application program or module, the number of times item-name or group-name occurs in the array is contained in the item *item-name-a*.

index-name is a name, conforming to the rules for member names, that is to be used as the index name when COBOL data descriptions are generated by the DataManager Source Language Generation facility. The *index-name* is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for *index-name* when the data definition in which it appears is encoded) so *index-name* cannot be interrogated and can be the same as another name, an alias, or a catalog classification in the data dictionary.

common-clauses are described in the *ASG-Manager Products Dictionary/Repository User's Guide*.

The IDMS-RECORD Data Definition

The IDMS-RECORD member defines an IDMS record type or an IDD record. All IDMS or IDD rules for Record definition must be complied with.

Common clauses can be present in any type of data definition statement. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

Clauses can be declared in any order, provided that subordinate clauses are not separated from the other elements of the clause of which they form a part.

IDMS-RECORD Dummy Records

A record containing the IDMS record data definition statement can be inserted into the data dictionary source dataset by a suitable command and an encoded record can subsequently be generated and inserted into the data entries dataset. If, when the encoded record is generated, any member whose name appears in the data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as:

- A dummy IDMS-SET record, if its name appears in a STORED VIA clause
- A dummy ITEM record, if its name appears in a STORED USING/BY clause, a STORED VSAM FILE KEY clause, a CONTAINS clause or a SEE clause,
- A dummy IDMS-AREA, if its name appears in an AREA clause,
- A dummy MODULE record, if its name appears in a CALL clause,
- A dummy FILE if in a STORED VSAM FILE clause.

The STORED and CALLS Clauses

The STORED clause specifies the Location Mode of records of this type; this can be any one of these:

- DIRECT
- VIA a particular IDMS-SET
- Calculated, using a particular KEY
- VSAM

The member named in the CALL clause defines a procedure module to which control will be transferred when certain IDMS data manipulation functions are performed on records of this type. The module is a form of user-exit in IDMS, to allow a record to be examined, validated, and modified on its way into or out of the database, or in case of error.

The CONTAINS Clause

The contents of the record are specified in the CONTAINS clause. The CONTAINS keyword is followed by a list of definitions of the successive parts of the record. Each part of the record is defined by a content declaration that may be conditional and/or may have alternative content declarations that also may be conditional; so that the definition of each part of the record comprises a content declaration and any associated ELSE clauses and/or IF clauses. If the record comprises two or more parts, the definition of each part except the last must be followed by a comma which can optionally be followed by spaces.

Any part of the record can be specified as having any number of alternative contents. The alternative content declarations are separated by the keyword ELSE. The alternative contents need not occupy the same amount of physical storage. The expression *ELSE clause* thus refers to:

```
ELSE content
```

Any content declaration can be specified as conditional, that is, as applying only if a stated condition or combination of conditions is satisfied. For a content declaration to be conditional, content must be immediately followed by an IF clause.

It follows that any part of the record can have alternative conditional contents, declared in this form:

```
content IF clause ELSE content IF clause
 [ELSE content IF clause]. . .
```

and that any combination of conditional and non-conditional alternative contents can be declared for any part of the record.

The ALIGNMENT Keyword

It is not meaningful to include any of the keywords ALIGNED, UNALIGNED, or NOT-ALIGNED in a content declaration that declares a group or an array of groups; the data definitions of the groups determine the alignment or non-alignment of the contained items.

ALIGNED is the equivalent of COBOL SYNCHRONIZED or PL/I ALIGNED. It means that the binary item(s) or floating point item(s) declared (as an individual item or as elements of an array) in the content declaration is/are aligned to half word, full word or double word boundaries, thus:

- Binary items having a length of four decimal digits or less occupy a complete half word
- Binary items having a length of from five to nine decimal digits occupy a full word
- Binary items having a length of from ten to eighteen decimal digits occupy two full words, but are not necessarily aligned to a double word boundary floating-point items having six digits or less in the mantissa occupy a full word
- Floating-point items having from seven to sixteen digits in the mantissa occupy a double word.

UNALIGNED means that the binary item(s) or floating point item(s) declared (as an individual item or as elements of an array) in the content declaration is/are not necessarily aligned to word or half word boundaries. (The amount of space occupied is the same as for ALIGNED items, but the positioning relative to word boundaries can differ.)

NOT-ALIGNED means the same as UNALIGNED.

A bit string item which is ALIGNED will begin on the next byte boundary. A bit string item which is UNALIGNED or NOT-ALIGNED will begin on the next available bit.

In a content declaration, the ALIGNED, UNALIGNED or NOT-ALIGNED element, the KNOWN-AS clause and the INDEXED-BY clause can, if applicable, be declared in any order; but they must not precede any of the other elements of the content declaration (though they must precede any associated ELSE clauses and/or IF clauses).

The IDMS-LOGICAL-RECORD Member Type

Syntax of the IDMS-LOGICAL-RECORD Member Type

```
IDMS-LOGICAL-RECORD

CONTAINS record-name [KNOWN-AS role-name]
       [,record-name [KNOWN-AS role-name]]...

[ERROR { CLEAR   }]
       { NOCLEAR }

[NOT-FOUND { CLEAR   }]
           { NOCLEAR }

[OBTAIN PATHS path-group [,path-group]...]
[MODIFY PATHS path-group [,path-group]...]
[STORE PATHS path-group [,path-group]...]
[ERASE PATHS path-group [,path-group]...]
[common-clauses]
{ ; }
{ . }
```

where:

record-name is the name of a member that is an IDMS-RECORD

role-name is a 1- to 16-character name

path-group is the name of a member that is an IDMS-PATH-GROUP

common-clauses are described in the *ASG-Manager Products Dictionary/Repository User's Guide*.

The IDMS-LOGICAL-RECORD Data Definition

This data definition statement defines a logical record that programs using a subschema can access.

The CONTAINS clause identifies all the records that participate as elements in the logical record.

The KNOWN-AS clause specifies the role-name which is used to provide a unique identifier to a record that occurs more than once in a single logical record. Role-name must be a name of 1 to 16 characters.

The OBTAIN, MODIFY, STORE, and ERASE PATHS clauses identify the IDMS-PATH-GROUP members which define the paths used to access the logical record in conjunction with each verb.

IDMS-LOGICAL-RECORD Dummy Records

A record containing the IDMS logical-record data definition statement can be inserted into the data dictionary source dataset by a suitable command and an encoded record can subsequently be generated and inserted into the data entries dataset. If, when the encoded record is generated, any member whose name appears in the data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as:

- A dummy IDMS-RECORD, if its name appears in a CONTAINS clause
- A dummy IDMS-PATH-GROUP, if its name appears in an OBTAIN, MODIFY, STORE or ERASE clause

The IDMS-PATH-GROUP Member Type

Syntax of the IDMS-PATH-GROUP Member Type

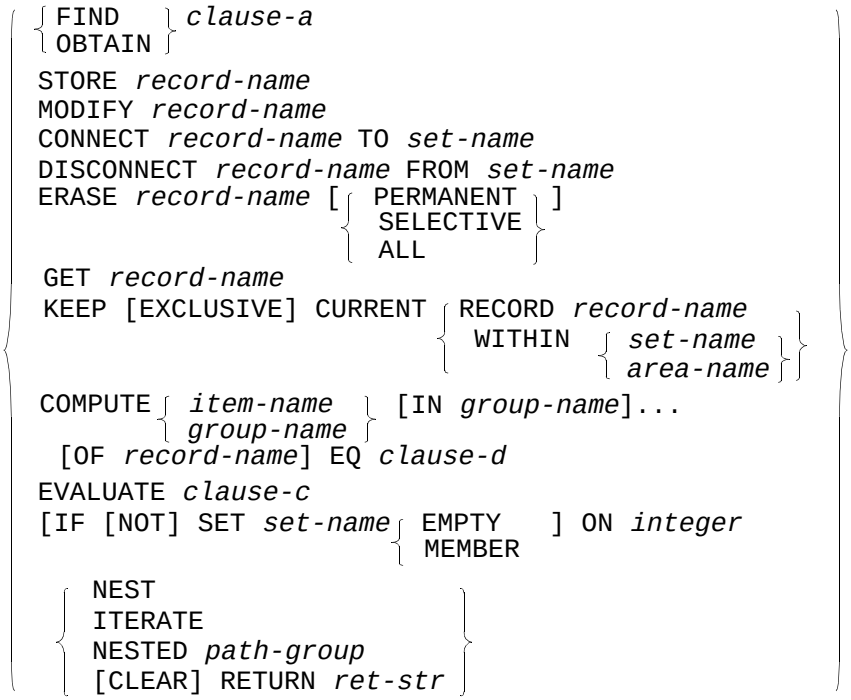
```
IDMS-PATH-GROUP
[FOR selection [,selection]...]
DO command [DO command]...
[common-clauses]
{ ; }
{ . }
```

where:

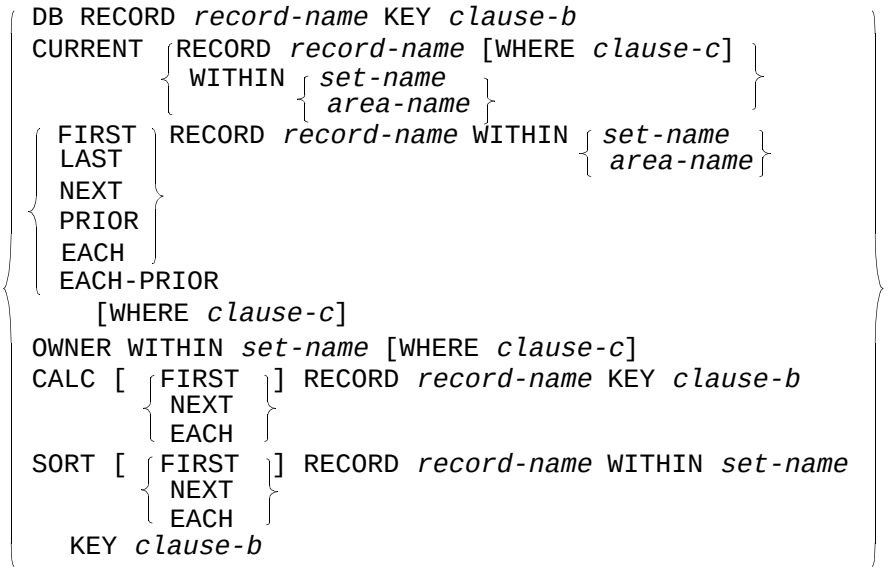
selection is:

```
{ KEYWORD name                                                    }
{ { FIELDNAME-EQ } { item-name  } [IN group-name]...              }
{ { FIELDNAME   } { group-name }                                  }
{   [OF record-name]                                              }
{ RECORD record-name                                              }
```

command is:



clause-a is:



clause-b is:

```
{ FIELD { item-name  } [IN group-name]...
        { group-name }
    [OF record-name  { LR      } ]
                     { REQUEST }
  'string'
  ARITHMETIC 'string' [CONTAINING { item-name  }
                                  { group-name }
                                [, { item-name  } ]...]
                                   { group-name }
[AND clause-c]
```

clause-c is:

```
[NOT] comp-a [ { AND } [NOT] comp-a]...
               { OR  }
```

comp-a is:

```
clause-d { CONTAINS } clause-d
         { MATCHES  }
         { EQ       }
         { NE       }
         { GT       }
         { LT       }
         { GE       }
         { LE       }
```

clause-d is:

```
{ FIELD { item-name  } [IN group-name]...
        { group-name }
    [OF record-name  { LR      } ]
                     { REQUEST }
  'string'
  ARITHMETIC 'string' [CONTAINING { item-name  }
                                  { group-name }
                                [, { item-name  } ]...]
                                   { group-name }
```

name is up to 32 characters

item-name is the name of a member that is an ITEM

group-name is the name of a member that is a GROUP

record-name is the name of a member that is an IDMS-RECORD

set-name is the name of a member that is an IDMS-SET

string is up to 16 alphanumeric characters

area-name is the name of a member that is an IDMS-AREA

integer is an unsigned integer of from 1 to 18 digits, being the number of times item-name or group-name occurs in the array

path-group is the name of a member that is an IDMS-PATH-GROUP

ret-str is a string of 1 to 16 alphanumeric characters

common-clauses are described in the *ASG-Manager Products Dictionary/Repository User's Guide*.

The IDMS-PATH-GROUP Data Definition

This data definition statement defines the paths used by the IDMS Logical Record Facility (LRF) to service program requests for access to logical records.

The FOR clause identifies selectors to be used as the basis of path selection to service logical-record requests. For this path to be chosen, the WHERE clause of the program request must supply information that matches all selectors specified in any one of the path's FOR clauses.

If the FOR clause is omitted, the path-group may be used only as a collection of path commands to be included as a nested block following an ON statement.

The DO statement specifies the path commands to be executed.

IDMS-PATH-GROUP Dummy Records

A record containing the IDMS path-group data definition statement can be inserted into the data dictionary source dataset by a suitable command and an encoded record can subsequently be generated and inserted into the data entries dataset. If, when the encoded record is generated, any member whose name appears in the data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as:

- A dummy ITEM, if its name appears in a FIELDNAME, FIELDNAMEEQ, or COMPUTE clause, or FIELD or ARITHMETIC CONTAINING subclause
- A dummy GROUP, if its name appears in an IN subclause
- A dummy IDMS-RECORD, if its name appears in an OF subclause, or a RECORD, STORE, MODIFY, CONNECT, DISCONNECT, ERASE, or GET clause
- A dummy IDMS-SET, if its name appears in a CONNECT TO, DISCONNECT FROM, SET, or WITHIN clause

The IDMS-VIEW Member Type

Syntax of the IDMS-VIEW Member Type

```
IDMS-VIEW
RECORD record-name [KNOWN-AS local-name]
[SELECTING { item-name  } [KNOWN-AS local-name]
           { group-name }
        [, { item-name  } [KNOWN-AS local-name] . . . ]
           { group-name }
[common-clauses]

 { ; }
 { . }
```

where:

record-name is a name of a member that is an IDMS-RECORD

local-name is a name, conforming to the rules for member names, that can be used instead of the name or alias of the *record-name*, *group-name* or *item-name* when IDMS source statements are generated from this data definition by the DataManager Source Language Generation facility. The *local-name* is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for *local-name* when the data definition in which it appears is encoded) so *local-name* cannot be interrogated and can be the same as another name, an alias, or a catalog classification in the data dictionary. The *local-name* is the name by which the contained member is known only within the view defined by this data definition.

item-name is the name of a member that is an ITEM

group-name is the name of a member that is a GROUP

common-clauses are described in the *ASG-Manager Products Dictionary/Repository User's Guide*.

The IDMS-VIEW Data Definition

This data definition statement defines a view of a record type that is used in an IDMS subschema; that is, those fields of the record type that are used by the subschema.

The IDMS-VIEW data definition defines a view of an IDMS record type which is used in an IDMS Subschema. All IDMS rules regarding the Data Definition Language for Subschema must be obeyed.

The RECORD clause identifies the IDMS-RECORD member of which the IDMS-VIEW is a subset.

When the SELECTING clause is not specified, whole record processing is implied. The SELECTING clause implies that only the specified groups (and their directly or indirectly contained groups and/or items) or items are processed.

Any direct or indirect reference from a SELECTING clause to an item is assumed to be the HELD-AS form of that item. If the item has no HELD-AS form, default assumptions are made as to the relevant form of the item, in the order DEFAULTED-AS, ENTERED-AS, REPORTED-AS. The form first encountered in this order is taken as the defaulted form. The version of the item is that specified or defaulted for it in the CONTAINS clause of the containing IDMS-RECORDS.

Common clauses can be present in any type of data definition statement. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

IDMS-VIEW Dummy Records

A record containing the IDMS view data definition statement can be inserted into the data dictionary source dataset by a suitable command and an encoded record can subsequently be generated and inserted into the data entries dataset. If, when the encoded member is generated, any member whose name appears in the data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as:

- A dummy IDMS-RECORD, if the member name appears in a RECORD clause
- A dummy ITEM, if the member name appears in a SELECTING clause or a SEE clause

The IDMS-SUBSCHEMA Member Type

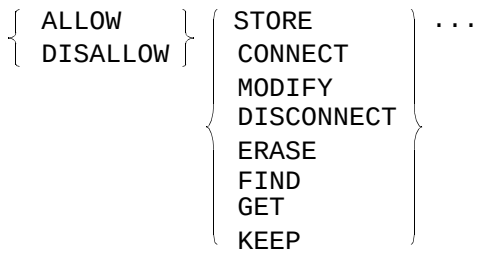
Syntax of the IDMS-SUBSCHEMA Member Type

```
IDMS-SUBSCHEMA
ACCESSES database-name
[AUTHORIZATION{ ON  } ]
              { OFF }
[USAGE { DML   } ]
       { MIXED }
       { LR    }
[STATISTICS TO subschema-name [OF database-name]
   [FOR program-name]]...
[LR-CURRENCY{ RESET   } ]
            { NORESET }

{ ALL-AREAS [OPTIONS subschema-area-options]                 }
{ AREA area-name [OPTIONS subschema-area-options]            }
{   [AREA area-name [OPTIONS subschema-area-options]]...     }

[RECORD record-name [KNOWN-AS local-name]
   [OPTIONS record-options] [PRIORITY{ NULL } ]
                                     { n    }
   [SELECTING{ item-name  } [KNOWN-AS local-name]
             { group-name }
          [, {item-name  } [KNOWN-AS local-name]]...]]...
             {group-name }
[VIEW view-name [OPTIONS record-options]
   [PRIORITY{ NULL } ]]...
            { n    }

{ ALL-SETS [OPTIONS set-options]                 }
{ SET set-name [OPTIONS set-options]             }
{    [SET set-name [OPTIONS set-options]]...     }

[LOGICAL-RECORD lr-name]...

DMCL dmcl-name
{ AS subschema-name                                                  }
{ I-O [BUFFER name  PAGES page SIZE size                             }
{      [CONTROL-INTERVAL c-i-size]                                   }
{       [VSAM-TYPE{ LSR } STRNO j{ KEYLEN k }]]...                   }
{                 { NSR }        { BUFNI m  }                       }
{     [JOURNAL-BUFFER name PAGES p SIZE s]...                        }
{      { ALL-AREAS [OPTIONS dmcl-area-options]                       }
{      { AREA area-name [OPTIONS dmcl-area-options]                  }
{        [AREA area-name [OPTIONS dmcl-area-options]]...             }
{     [JOURNAL TAPE-FILE file-name  BLOCK-SIZE t-size                }
{                                    BUFFER name                     }
{           [ASSIGN external-name] [DEVICE device]                   }
{           [DISKS BLOCK-SIZE d-size BLOCK-COUNT count               }
{            FILE file-name [ASSIGN external-name]                   }
{                                    [DEVICE device]                 }
{             [FILE file-name [ASSIGN external-name]                 }
{                             [DEVICE device]]...]]                  }

[common-clauses]
{ ; }
{ . }
```

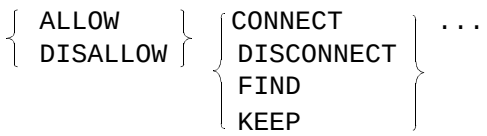
where:

subschema-area-options is:

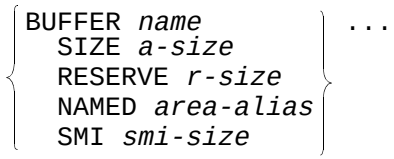
record-options is:



set-options is:



dmcl-area-options is:



The IDMS-SUBSCHEMA Data Definition

The IDMS-SUBSCHEMA data definition defines an IDMS Subschema and its associated Device Media Control Language specification (DMCL). All IDMS rules regarding the Data Definition Language for Subschema and DMCL must be obeyed.

RECORD clauses specify which records, groups, and data-items are to be processed. When the SELECTING subordinate clause is not specified, whole record processing is implied. The SELECTING subordinate clause implies that only the specified groups (and their directly or indirectly contained groups and/or items) or items are processed.

Any direct or indirect reference from a SELECTING clause to an item is assumed to be to the HELD-AS form of that item. If the item has no HELD-AS form, default assumptions are made as to the relevant form of the item, in the order DEFAULTED-AS, ENTERED-AS, REPORTED-AS. The form first encountered in this order is taken as the defaulted form. The version of the item is that specified or defaulted for it in the CONTAINS clause of the containing IDMS-RECORD.

VIEW clauses specify which IDMS-VIEW members are to be processed. They thus provide an alternative way to RECORD clauses of specifying (indirectly) the records, or the groups and items within records, that are to be processed.

For each RECORD clause or VIEW clause, record-options can specify the processing options selected. For any RECORD clause or VIEW clause for which record-options are not specified, it is assumed that all processing options are allowed.

The LOGICAL-RECORD clause identifies the IDMS-LOGICAL-RECORDs that programs using the subschema can access.

Common clauses can be present in any type of data definition statement. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

IDMS-SUBSCHEMA Dummy Records

A record containing the IDMS-Subschema data definition statement can be inserted into the data dictionary source dataset by suitable command and an encoded record can subsequently be generated and inserted into the data entries dataset. If, when the encoded record is generated, any member whose name appears in the data definition statement has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as:

- A dummy IDMS-SUBSCHEMA, if the member name appears in a STATISTICS TO clause, or in a DMCL clause that has an AS subschema name subordinate clause
- A dummy IDMS-AREA, if the member name appears in an AREA clause (of a DMCL or Subschema definition)
- A dummy FILE, if the member name appears in a JOURNAL clause
- A dummy IDMS-RECORD, if the member name appears in a RECORD clause
- A dummy IDMS-VIEW, if the member name appears in a VIEW clause
- A dummy ITEM, if the member name appears in a SELECTING clause or a SEE clause
- A dummy IDMS-SET, if the member name appears in a SET clause
- A dummy IDMS-LOGICAL-RECORD, if the member name appears in a LOGICAL-RECORD clause

The ACCESSES Clause

ALL-AREAS specifies that all areas defined for the database named in the ACCESSES clause are processed. The *subschema-area-options* specify the processing options selected.

AREA clauses specify the areas processed, where not all the areas defined for the database named in the ACCESSES clause are processed. For each area processed, *subschema-area-options* can specify the processing options selected.

ALL-SETS specifies that all sets defined for the database named in the ACCESSES clause are processed. The *set-options* specify the processing options selected; if *set-options* is omitted, it is assumed that all processing options are allowed.

SET clauses specify the sets processed, where not all the sets defined for the database named in the ACCESSES clause are processed. For each set processed, *set-options* can specify the processing options selected. For any processed set for which *set-options* is not specified, it is assumed that all processing options are allowed.

The DMCL Clause

The DMCL clause can contain a full definition of the Device Media Control Language specification, or can refer (by an AS subschema-name clause) to another member in which the full definition is contained. Thus, if a DMCL specification can apply to many Subschema, it can be defined in conjunction with each Subschema, or can be defined in one member only, to which other members requiring the same DMCL specification can refer.

The DISKS subordinate clause within the JOURNAL clause of the DMCL specification is only to be included if the journal is maintained on both disk and tape. In this case, one subordinate file clause is included for each disk file in use. If the journal is being maintained on tape alone, the DISKS clause is omitted.

The DEVICE clause is required for DOS files, but is optional for OS unless VSAM is specified.

When DMCL specifications are generated by the Source Language Generation facility, if two or more BLOCK-COUNT subordinate clauses are present in the DISKS clause, the first one encountered is taken to generate the FILE CONTAINS statements, and subsequent ones are ignored.

If the JOURNAL subordinate clause is omitted, the DMCL clause must come later in the data definition than the ALL-AREAS clause or AREA clauses. Subject to this restriction, the ACCESSES clause, the ALL-AREAS clause or the AREA clauses, the RECORD clauses, the ALL-SETS clause or the SETS clauses, and the DMCL clause can be in any order.

SYSTEM, PROGRAM, and MODULE Data Definition for an IDMS Environment

Introduction

For the IDMS Interface, the PROCESSES clause is included in the format of SYSTEM, PROGRAM, and MODULE statements to specify which processing views of the database (that is, which IDMS Subschemas) are relevant to the member.

The PROCESSES clause is used to specify the names of the IDMS subschema members to which this SYSTEM, PROGRAM, or MODULE relates.

A record containing the data definition statement of the system, program, or module that includes the PROCESSES clause can be inserted into the data dictionary's source dataset by a suitable command and an encoded record can subsequently be generated and inserted into the data entries dataset. If, when the encoded record is generated, any member whose name appears in the PROCESSES clause has no data entries record, DataManager creates a dummy data entries record for that member. The dummy record is created as a dummy IDMS-SUBSCHEMA.

Syntax of the IDMS-PROCESSES Clause

```
PROCESSES IDMS SUBSCHEMAS
  subschema-name [KNOWN-AS procedure-name]
[,subschema-name [KNOWN-AS procedure-name]]...
```

where:

subschema-name is the name of a member that is an IDMS-SUBSCHEMA

procedure-name is a name not more than eight characters in length, that both conforms to the rules for member names and is a legal CSECT name or entry point for input into IBM OS or DOS assemblers. The procedure-name can be used instead of the *process-name* member name or alias when IDMS source language is generated from this data definition by the DataManager Source Language Generation facility. The *procedure-name* is not separately recorded in the data dictionary (that is, no dummy data entries record and no index record is created for *procedure-name* when the data definition in which it appears is encoded) so *procedure-name* cannot be interrogated and can be the same as another name, an alias or a catalog classification in the data dictionary. The *procedure-name* is the name by which *process-name* is known only in the context of this data definition.

5 DataManager Correspondence Tables

Schema Relationship Table

Correspondence Between IDMS Schema Data Description Language and DataManager Data Definition

IDMS Schema Syntax	DataManager Syntax
ADD SCHEMA NAME *is schema name*	*database-name* IDMS-DATABASE
ASSIGN RECORD IDS FROM *i*	RECORD-ID-START *i*
PAGE CONTAINS *j* RECORDS MAXIMUM	MAXIMUM-RECORDS-PER-PAGE *j*
PAGE GROUP IS *k*	PAGE GROUP *k*
ADD FILE NAME IS JOURNAL	JOURNAL
ADD FILE NAME IS *file-name*	DATASET *file-name*
ASSIGN TO *external-name*	ASSIGN *external-name*
DEVICE TYPE IS *device-type*	DEVICE *device*
ADD AREA NAME IS *area-name*	AREAS *area-name else area-name* IDMS-AREA
PAGE RANGE IS *pn-1* { THRU | FOR } *pn-2*	PAGES *page-1* { TO | FOR } *page-2*
WITHIN FILE *file-name*	{ OF | IN } *file-name*
FROM *sbn* { THRU *ebn* | FOR { ALL | *bcn* } }	BLOCK *block* { TO *block* | FOR { ALL | *block* } }
CALL *proc-name* { BEFORE | AFTER | ERROR }	CALL *process-name* { BEFORE | AFTER | ERROR }
{ READY FOR { EXCLUSIVE | PROTECTED | SHARED } { UPDATE | RETRIEVAL } | FINISH | COMMIT | ROLLBACK }	{ READY FOR { EXCLUSIVE | PROTECTED | SHARED } { UPDATE | RETRIEVAL } | FINISH | COMMIT | ROLLBACK }

Correspondence Between IDMS Schema Data Description Language and DataManager Data Definition

IDMS Schema Syntax	DataManager Syntax
ADD RECORD NAME IS *record-name*	*Indirectly via* IDMS-SET *else* STAND-ALONE *record-name else record-name* IDMS-RECORD
RECORD ID IS *record-id*	IDENTITY *record-id*
LOCATION MODE IS CALC USING [*calc-elem . . .*]	STORED {USING | BY} KEY {*item-name* [*version*] | *group-name*}
DUPLICATES {FIRST | LAST | NOT ALLOWED}	DUPLICATES {FIRST | LAST | DISALLOWED}
VIA *set-name* SET	VIA *set-name*
DISPLACEMENT *pc* PAGES	DISPLACED *page-count*
DIRECT	DIRECT
VSAM CALC FROM *file-name* USING *elem-name*	VSAM FILE *file-name* {USING | BY} KEY
DUPLICATES ARE {NOT ALLOWED | UNORDERED}	DUPLICATES {DISALLOWED | UNORDERED}
VSAM-TYPE IS {FIXED | VARIABLE}	VSAM-TYPE IS {FIXED | VARIABLE}
LENGTH {SPANNED | NONSPANNED}	LENGTH {SPANNED | UNSPANNED}
WITHIN *area-name* AREA FROM *spn* THRU *epn* OFFSET *n* {PERCENT | PAGES} FOR *pn* {PERCENT | PAGES}	AREA *area-name* FROM *page-1* TO *page-2* OFFSET {PERCENT *i* | PAGES *page-3*} FOR {PERCENT *k* | PAGES *page-4*}
MINIMUM ROOT LENGTH IS {CONTROL LENGTH | *root-length* CHARACTERS | RECORD LENGTH}	MINIMUM ROOT {CONTROL | *root-length* | RECORD}
MINIMUM FRAGMENT LENGTH IS {*fragment-length* CHARACTERS | RECORD LENGTH}	MINIMUM FRAGMENT {*fragment-length* | RECORD}

Correspondence Between IDMS Schema Data Description Language and DataManager Data Definition

IDMS Schema Syntax	DataManager Syntax
CALL *proc-name* { BEFORE | AFTER | ON ERROR DURING }	CALL *procedure-name* { BEFORE | AFTER | ERROR }
{ STORE | CONNECT | MODIFY | DISCONNECT | ERASE | FIND | GET | RETURN }	{ STORE | CONNECT | MODIFY | DISCONNECT | ERASE | FIND | GET | RETURN }
level-n element-name . . . SYNONYM NAME FOR *alias-type* IS *alias*	CONTAINS *contents* (as for COBOL generation) ALIAS *alias-type alias*
{ ASCENDING | DESCENDING } KEY IS *key-elem*	KEYS { *item-name* | *group-name* } { ASCENDING | DESCENDING }
ADD SET NAME IS *set-name*	SETS *set-name else* *set-name* IDMS-SET
ORDER IS { FIRST | LAST | NEXT | PRIOR | SORTED }	ORDER { FIRST | LAST | NEXT | PRIOR | SORTED }
MODE IS CHAIN [LINKED TO PRIOR] VSAM INDEX FROM FILE *file-name* INDEX BLOCK CONTAINS *kcn* KEYS DISPLACEMENT *n* PAGES	MODE CHAIN [LINKED PRIOR] VSAM INDEX *file-name* INDEXED KEYS *p* DISPLACED *page-count*
OWNER IS *record-name* NEXT DBKEY POSITION { AUTO | *int1* } PRIOR DBKEY POSITION IS { AUTO | *int2* }	OWNER *record-name* NEXT POSITION { AUTO | *position* } PRIOR-POSITION { AUTO | *position* }

Correspondence Between IDMS Schema Data Description Language and DataManager Data Definition

IDMS Schema Syntax	DataManager Syntax
OWNER IS SYSTEM WITHIN AREA *area-name* FROM *spn* THRU *epn* OFFSET *n* { PERCENT | PAGES } FOR *pn* { PERCENT | PAGES }	OWNED-BY SYSTEM AREA *area-name* FROM *page-1* TO *page-2* OFFSET { PERCENT *i* | PAGES *page-3* } FOR { PERCENT *s* | PAGES *page-4* }
MEMBER IS *record-name* NEXT DBKEY POSITION IS { AUTO | *int1* } PRIOR DBKEY POSITION IS { AUTO | *int2* } LINKED TO OWNER OWNER DBKEY POSITION IS { AUTO | *int3* } INDEX DBKEY POSITION IS { AUTO | *int4* }	MEMBER *record-name* NEXT-POSITION { AUTO | *position* } PRIOR-POSITION { AUTO | *position* } OWNER-POSITION { AUTO | *position* } INDEX-POSITION { AUTO | *position* }
{ MANDATORY | OPTIONAL | AUTOMATIC | MANUAL }	{ MANDATORY | OPTIONAL | AUTOMATIC | MANUAL }
{ ASCENDING | DESCENDING } KEY IS { *elem-name* | DBKEY } { COMPRESSED | UNCOMPRESSED }	SORT KEY IS *item-name* [*version*] DBKEY { ASCENDING | DESCENDING | COMPRESSED | UNCOMPRESSED }
DUPLICATES ARE { FIRST | LAST | NOT ALLOWED | UNORDERED }	DUPLICATES ARE { FIRST | LAST | DISALLOWED | UNORDERED }

Device Media Control Language Relationship Table

Correspondence Between IDMS Device Media Control Language (DMCL) and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS Syntax	DataManager Syntax
DEVICE-MEDIA DESCRIPTION.	
DEVICE-MEDIA NAME IS *dmcl-name* OF SCHEMA NAME *schema-name*	DMCL *dmcl-name* ACCESSES *database-name*
BUFFER SECTION.	
BUFFER NAME IS *buffer* PAGE CONTAINS *psn* CHARACTERS BUFFER CONTAINS *pcn* PAGES CONTROL INTERVAL CONTAINS *int* CHARACTERS VSAM TYPE IS {LSR | NSR} STRNO IS *sn* KEYLEN IS *kl* BUFNI IS *bn* JOURNAL BUFFER NAME IS *j-name* PAGE CONTAINS *psn* CHARACTERS BUFFER CONTAINS *pcn* PAGES	BUFFER *name* SIZE *size* PAGES *page* CONTROL-INTERVAL *c-i-size* VSAM-TYPE {LSR | NSR} STRNO *j* KEYLEN *k* BUFNI *m* JOURNAL-BUFFER *name* SIZE *s* PAGES *p*
AREA SECTION.	
COPY *area-name* AREA	{ALL-AREAS | AREA *area-name*}
BUFFER IS *buffer-name* PAGE CONTAINS *psn* CHARACTERS PAGE RESERVE CONTAINS *rsn* CHARACTERS ALIAS *new-name* SMI BASED ON *opsn* CHARACTERS	BUFFER *name* SIZE *a-size* RESERVE *r-size* NAMED *area-alias* SMI *smi-size*
JOURNAL SECTION.	
JOURNAL BLOCK CONTAINS *bsn* CHARACTERS JOURNAL BUFFER IS *buffer-name* FILE CONTAINS *bcn* BLOCKS FILE NAME IS *disk-file-name* ASSIGN TO *external-name* DEVICE TYPE IS *device* ARCHIVAL JOURNAL BLOCK CONTAINS *bsn* CHARACTERS FILE NAME IS *tape-file-name* ASSIGN TO *external-name* DEVICE TYPE IS *device*	JOURNAL. . .BLOCK-SIZE *t-size* JOURNAL. . .BUFFER *name* DISKS BLOCK-COUNT *bcn* DISKS. . .FILE *file-name* ASSIGN *external-name* DEVICE *device* TAPE-FILE. . .BLOCK-SIZE *d-size* TAPE-FILE *file-name* ASSIGN *external-name* DEVICE *device*

Subschema Relationship Table

Correspondence Between IDMS Subschema Data Descriptions Language and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS SUBSCHEMA Syntax	DataManager Syntax
ADD SUBSCHEMA NAME IS *subschema-name*	*subschema-name* IDMS-SUBSCHEMA
OF SCHEMA NAME IS *schema*	ACCESSES *database-name*
DMCL NAME IS *dmcl-name*	DMCL *dmcl-name*
AUTHORIZATION IS { ON | OFF }	AUTHORIZATION IS { ON | OFF }
USAGE IS { DML | MIXED | LR }	USAGE IS { DML | MIXED | LR }
TRANSFER STATISTICS TO SUBSCHEMA NAME *subschema* OF SCHEMA NAME IS *schema* FOR PROGRAM NAME IS *prog*	STATISTICS TO *subschema-name* OF *database-name* FOR *program-name*
LR CURRENCY { RESET | NO RESET }	LR-CURRENCY { RESET | NO RESET }
ADD AREA NAME IS *area-name*	{ ALL AREAS | AREA *area-name* }
{ SHARED | PROTECTED | EXCLUSIVE }	{ SHARED | PROTECTED | EXCLUSIVE }
{ UPDATE | RETRIEVAL }	{ UPDATE | RETRIEVAL }
IS { ALLOWED | NOT ALLOWED }	{ ALLOW | DISALLOW }
DEFAULT USAGE IS	DEFAULT-USAGE
{ SHARED | PROTECTED | EXCLUSIVE }	{ SHARED | PROTECTED | EXCLUSIVE }
{ UPDATE | RETRIEVAL }	{ UPDATE | RETRIEVAL }

Correspondence Between IDMS Subschema Data Descriptions Language and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS SUBSCHEMA Syntax	DataManager Syntax
ADD RECORD NAME IS *record*	indirectly via IDMS-VIEW *else* RECORD *record-name*
VIEW ID IS *view-name*	VIEW *view-name*
{ STORE CONNECT MODIFY DISCONNECT ERASE FIND GET KEEP }	{ STORE CONNECT MODIFY DISCONNECT ERASE FIND GET KEEP }
IS { ALLOWED NOT ALLOWED }	{ ALLOW DISALLOW }
ELEMENTS ARE { ALL *field*. . . }	SELECTING { *item-name* *group-name* }
PRIORITY { NULL *p* }	PRIORITY { NULL *n* }
ADD SET NAME IS *set-name*	{ ALL-SETS SET *set-name* }
{ CONNECT DISCONNECT FIND KEEP } IS { ALLOWED NOT ALLOWED }	{ ALLOW DISALLOW } { CONNECT DISCONNECT FIND KEEP }
ADD LOGICAL RECORD NAME IS *lrec-name*	LOGICAL-RECORD *lr-name* *lr-name* IDMS-LOGICAL-RECORD
ELEMENTS ARE *record-name*	CONTAINS *record-name*
ROLE IS *role-name*	KNOWN-AS *role-name*
ON LR-ERROR { CLEAR NOCLEAR }	ERROR { CLEAR NOCLEAR }
ON LR-NOT-FOUND { CLEAR NOCLEAR }	NOT-FOUND { CLEAR NOCLEAR }
ADD PATH-GROUP NAME-IS { OBTAIN MODIFY STORE ERASE } *lrec-name*	{ OBTAIN PATHS *path-group* MODIFY PATHS *path-group* STORE PATHS *path-group* ERASE PATHS *path group* } else *path-group* IDMS-PATH-GROUP
SELECT FOR	FOR
KEYWORD *name*	KEYWORD *name*
FIELDNAME-EQ *field*	FIELDNAME-EQ { *item-name* *group-name* }
OF *group* OF *record*	IN *group-name* OF *record-name*
FIELDNAME *field*	FIELDNAME { *item-name* *group-name* }
OF *group* OF *record*	IN *group-name* OF *record-name*

Correspondence Between IDMS Subschema Data Descriptions Language and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS SUBSCHEMA Syntax	DataManager Syntax
ELEMENT *record* {FIND | OBTAIN} KEEP EXCLUSIVE *record* WHERE DBKEY	RECORD *record-name* {FIND | OBTAIN} KEEP EXCLUSIVE DB RECORD *record-name*
EQ { *field* | OF *group* | OF *record* | OF LR | OF REQUEST | *literal* | *arith-exp* }	KEY { FIELD { *item-name* | *group-name* } | IN *group-name* | IN *record-name* | LR | REQUEST | '*string*' | ARITHMETIC '*arith-exp*' }
AND [NOT] { *field* | OF *group* | OF *record* | OF LR | *literal* | *arith-exp* }	AND [NOT] { FIELD { *item-name* | *group-name* } | IN *group-name* | OF *record-name* | LR | '*string*' | ARITHMETIC '*arith-exp*' }
CONTAINS MATCHES EQ NE GT LT GE LE	CONTAINS MATCHES EQ NE GT LT GE LE
{ *field* | OF *group* | OF *record* | OF LR | *literal* | *arith-exp* } {AND | OR} [NOT] . . .	{ FIELD { *item-name* | *group-name* } | IN *group-name* | OF *record-name* | LR | '*string*' | ARITHMETIC '*arith-exp*' } {AND | OR} [NOT] . . .

Correspondence Between IDMS Subschema Data Descriptions Language and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS SUBSCHEMA Syntax	DataManager Syntax
CURRENT { *record* / WITHIN *set* / WITHIN *area* }	CURRENT { RECORD *record-name* / WITHIN { *set-name* / *area-name* } }
WHERE [NOT]	WHERE [NOT]
{ *field* / OF *group* / OF *record* / OF LR / *literal* / *arith-exp* }	{ FIELD { *item-name* / *group-name* } / IN *group-name* / OF *record-name* / LR / '*string*' / ARITHMETIC '*arith-exp*' }
CONTAINS	CONTAINS
MATCHES	MATCHES
EQ	EQ
NE	NE
GT	GT
LT	LT
GE	GE
LE	LE
{ *field* / OF *group* / OF *record* / LR / *literal* / *arith-exp* }	{ FIELD { *item-name* / *group-name* } / IN *group-name* / OF *record-name* / LR / '*string*' / ARITHMETIC '*arith-exp*' }
{ AND / OR } [NOT]	{ AND / OR } [NOT]
. . .	. . .
{ FIRST / LAST / NEXT / PRIOR / EACH / EACH PRIOR } *record*	{ FIRST / LAST / NEXT / PRIOR / EACH / EACH-PRIOR } RECORD *record-name*
WITHIN { *set* / *area* }	WITHIN { *set-name* / *area-name* }
WHERE [NOT]	WHERE [NOT]
{ *field* / OF *group* / OF *record* / OF LR / *literal* / *arith-exp* }	{ FIELD { *item-name* / *group-name* } / IN *group-name* / OF *record-name* / LR / '*string*' / ARITHMETIC '*arith-exp*' }

Correspondence Between IDMS Subschema Data Descriptions Language and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS SUBSCHEMA Syntax	DataManager Syntax
CONTAINS MATCHES EQ NE GT LT GE LE { *field* { OF *group* { OF *record* { OF LR { *literal* { *arith-exp* { AND } [NOT] { OR } . . .	CONTAINS MATCHES EQ NE GT LT GE LE { FIELD {*item-name* { {*group-name* { IN *group-name* { OF *record-name* { LR { '*string*' { ARITHMETIC '*arith-exp*' { AND } [NOT] { OR } . . .
OWNER WITHIN *set*	OWNER WITHIN *set-name*
WHERE [NOT} { *field* { OF *group* { OF *record* { OF LR { *literal* { *arith-exp*	WHERE [NOT} { FIELD {*item-name* { {*group-name* { IN *group-name* { OF *record-name* { LR { '*string*' { ARITHMETIC '*arith-exp*'
CONTAINS MATCHES EQ NE GT LT GE LE { *field* { OF *group* { OF *record* { OF LR { *literal* { *arith-exp* { AND } [NOT] { OR } . . .	CONTAINS MATCHES EQ NE GT LT GE LE { FIELD {*item-name* { {*group-name* { IN *group-name* { OF *record-name* { LR { '*string*' { ARITHMETIC '*arith-exp*' { AND } [NOT] { OR } . . .

Correspondence Between IDMS Subschema Data Descriptions Language and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS SUBSCHEMA Syntax	DataManager Syntax
[FIRST] record [NEXT] [EACH]	CALC [FIRST] RECORD record-name [NEXT] [EACH]
WHERE CALCKEY EQ	KEY
field { OF group OF record OF LR OF REQUEST literal arithmetic-expression }	FIELD { item-name group-name } { IN group-name OF record-name LR REQUEST 'string' ARITHMETIC 'arithmetic-expression' }
AND [NOT]	AND [NOT]
{ field OF group OF record OF LR literal arithmetic-expression }	{ FIELD item-name group-name IN group-name OF record-name LR 'string' ARITHMETIC 'arithmetic-expression' }
CONTAINS MATCHES EQ NE GT LT GE LE	CONTAINS MATCHES EQ NE GT LT GE LE
{ field OF group OF record OF LR literal arithmetic-expression }	{ FIELD { item-name group-name } IN group-name OF record-name LR 'string' ARITHMETIC 'arithmetic-expression' }
{ AND OR } [NOT]	{ AND OR } [NOT]
. . .	. . .
[FIRST] record	SORT [FIRST] RECORD record-name
[NEXT]	[NEXT]
[EACH]	[EACH]
WITHIN set	WITHIN set-name
WHERE SORTKEY EQ	KEY
{ field OF group OF record OF LR literal arithmetic-expression }	{ FIELD { item-name group-name } IN group-name OF record-name LR 'string' ARITHMETIC 'arithmetic-expression' }

Correspondence Between IDMS Subschema Data Descriptions Language and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS SUBSCHEMA Syntax	DataManager Syntax

AND [NOT]
{ *field*
  OF *group*
  OF *record*
  LR
  OF REQUEST
literal
arith-exp }
	AND [NOT]
	{ FIELD { *item-name* | *group-name* }
	  IN *group-name*
	  OF *record-name*
	  LR
	  REQUEST
	'*string*'
	ARITHMETIC '*arith-exp*' }

CONTAINS
MATCHES
EQ
NE
GT
LT
GE
LE
{ *field*
  OF *group*
  OF *record*
  LR
literal
arith-exp }
{ AND | OR } [NOT]
	CONTAINS
	MATCHES
	EQ
	NE
	GT
	LT
	GE
	LE
	{ FIELD { *item-name* | *group-name* }
	  IN *group-name*
	  OF *record-name*
	  LR
	'*string*'
	ARITHMETIC '*arith-exp*' }
	{ AND | OR } [NOT]

. . . STORE *record* MODIFY *record* CONNECT *record* TO *set* DISCONNECT *record* FROM *set*	. . . STORE *record-name* MODIFY *record-name* CONNECT *record-name* TO *set-name* DISCONNECT *record-name* FROM *set-name*
ERASE *record* { PERMANENT | SELECTIVE | ALL } MEMBERS	ERASE *record-name* { PERMANENT | SELECTIVE | ALL }

GET *record*
KEEP EXCLUSIVE SUMMARY
{ *record* | WITHIN *set* | WITHIN *area* }
IF *set* IS [NOT] EMPTY
IF [NOT] *set* MEMBER
COMPUTE
{ *field* | OF *group* | OF *record* }
=
	GET *record-name*
	KEEP EXCLUSIVE CURRENT
	{ RECORD *record-name* | WITHIN { *set-name* | *area-name* } }
	IF [NOT] SET *set-name* { EMPTY | MEMBER }
	COMPUTE { *item-name* | *group-name* }
	  IN *group-name*
	  OF *record-name*
	EQ

Correspondence Between IDMS Subschema Data Descriptions Language and DataManager IDMS-SUBSCHEMA Data Definitions

IDMS SUBSCHEMA Syntax	DataManager Syntax
{ *field* | OF *group* | OF *record* | OF LR | *literal* | *arith-exp* }	{ FIELD { *item-name* | *group-name* } | IN *group-name* | OF *record-name* | LR | '*string*' | ARITHMETIC '*arith-exp*' }
EVALUATE [NOT] { *field* | OF *group* | OF *record* | OF LR | *literal* | *arith-exp* }	EVALUATE [NOT] { FIELD { *item-name* | *group-name* } | IN *group-name* | OF *record-name* | LR | '*string*' | ARITHMETIC '*arith-exp*' }
CONTAINS MATCHES EQ NE GT LT GE LE	CONTAINS MATCHES EQ NE GT LT GE LE
{ *field* | OF *group* | OF *record* | OF LR | *literal* | *arith-exp* } { AND | OR } [NOT]	{ FIELD { *item-name* | *group-name* } | IN *group-name* | OF *record-name* | LR | '*string*' | ARITHMETIC '*arith-exp*' } { AND | OR } [NOT]
. . . ON err-int { NEXT | ITERATE | DO *nested-block* END | [CLEAR] RETURN *str* }	. . . ON *integer* { NEXT | ITERATE | NESTED *path-group* | [CLEAR] RETURN *ret-str* }

Note:

Element descriptions may also be generated from the SELECTING clause of an IDMS-VIEW.

IDD Record Relationship Table

Correspondence Between IDD Records and DataManager Data Definitions

IDD Record Syntax	DataManager Syntax
ADD RECORD NAME IS *record-name* ENTITY-TYPE IS RECORD RECORD STORAGE IS *string* FORMAT IS *form* OCCURENCES ARE *ocn*	*record-name* IDMS-RECORD STORAGE *string* *form* OCCURENCES *p*

Index